INFORMATION RETRIEVAL PROJECT REPORT

Jaypee Institute of Information Technology, Noida

TOPIC: Sentimental Analysis on social media post

Submitted To Supervisor: Dr. Neetu Sardana

Submitted By:

GROUP NO: G14

Enrollment No	Name	Batch
19103070	Megha Agarwal	B2
19103206	Saloni Bansal	B6
19104004	Patil Amit Gurusidhappa	B11
19104007	Sanjoli Goyal	B11

Table of Contents

Acknowledgement

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highest respected and esteemed guide Dr. Neetu Sardana for her valuable guidance, encouragement and help for completing this work. Her useful suggestions for this whole work and cooperative behaviour are sincerely acknowledged. We would like to express our sincere thanks to her for giving us this opportunity to undertake this project. We would also like to express our indebtedness to our parents as well as our family members whose blessings and support always helped us to face the challenges ahead.

Name: Megha Agarwal
Enrolment Number: 19103070
Batch: B2

Name: Saloni Bansal
Enrolment Number: 19103206
Batch: B6

Name: Patil Amit Gurusidhappa
Enrolment Number: 19104004
Batch: B11

Name: Sanjoli Goyal
Enrolment Number: 19104007
Batch: B11

Abstract

Sentiment analysis is an emerging trend nowadays to understand people's sentiments in multiple situations in their quotidian life. Social media data would be utilised for the entire process i.e. the analysis and classification processes and it consists of text data and emoticons, emojis, etc. Many experiments were conducted in the antecedent studies utilising Binary and Ternary Classification whereas Multi-class Classification gives more precise and precise Classification. In Multi-class Classification, the data would be divided into multiple subclasses predicated on the polarities. Machine Learning and Deep Learning Techniques would be utilised for the classification process. Utilising Social media, sentiment levels can be monitored or analysed. In this project our main aim is to develop an efficient approach for analysing depressive tweets using various Machine Learning Algorithms like vector space model, tf-idf, naive Bayes etc.

Introduction

There are more than 264 million individuals worldwide who are suffering from depression. Depression is the main cause of disability worldwide and is a significant supporter of the overall global burden of disease.

Internet-based life gives the main edge chance to change early melancholy mediation services, especially in youthful grown-ups. Consistently, roughly 6,000 Tweets are tweeted on Twitter, which relates to more than 350,000 tweets sent for each moment, 500 million tweets for every day, and around 200 billion tweets for each year.

As indicated by the Pew Research Center, 72% of the public uses some sort of internet- based life.

The idea was to create a project that works to capture and analyze linguistic markers associated with the onset and persistence of depressive symptoms in order to build an algorithm that can effectively predict depression. To move forward the idea that linguistic markers in Tweets can be used to construct statistical models that can detect and even predict depression in ways that can complement and extend traditional approaches to diagnosis.

This project works to expand the scope of social media-based mental health measures and use existing research that has proven the correlation between depression and specific linguistic features in order to build an algorithm that can predict text-based signs of depression.

The use of linguistic markers as a tool in the analysis and diagnosis of depression has enormous potential. Depression can so quickly be seen in text, even without the use of complex models. Simply by collecting, cleaning, and processing available data, visual analysis alone can illuminate the difference between random Tweets and Tweets that have depressive characteristics.

The potential of linguistic analysis in the arena of mental health cannot be overstated. By analysing a person’s words, you have a clear and valuable window into his or her mental state.

Even the simplest analysis of social media can provide us with unprecedented access into individuals' thoughts and feelings and lead to substantially greater understanding and treatment of mental health.

Problem Statement

In this project our main aim is to develop an efficient approach for analysing depressive tweets using various Machine Learning Algorithms like vector space model, tf-idf, naive Bayes etc.

Literature Survey

[1] Yusuf Arslan, et.al (2017) used dynamic dictionaries and models in order to deploy several sentiment analysis techniques . Experiments are conducted on the small sized however relevant datasets using these techniques such that the popularity of particular terms can be understood along with the opinions of users related to them. Enhancement is seen as per the simulation results achieved through these experiments.

[2] Jaishree Ranganathan, et.al (2017) proposed a novel Spark system which utilised Specific Action Rule discovery based on Grabbing Strategy (SARGS) within its implementation . The complete Action Rules such as system DEAR, ARED and Association Action Rules are extracted with the help of this proposed system. The data is partitioned such that multiple nodes can access it for extracting their own Action Rules through this approach.

[3] Ankit Kumar Soni, (2017) proposed a novel system which utilised classification techniques for filtering out the important information from raw data available. Through this approach, the sentiments present within the twitter micro blogging services are analysed. The results of the proposed technique are compared with the results achieved after evaluating other existing approaches. As per the comparison analysis, it is seen that the proposed technique that involves maximum entropy classifier provides better results by providing around 74% of accuracy.

Methodology

1. Data Collection
2. Data Cleaning
3. Data Pre-processing
4. Training Dataset Formation
5. Model Building
6. Prediction on testing dataset

IR Model used in Project

<u>Vectorization our data set using TF-IDF</u>

Machine learning algorithms often use numerical data, so when dealing with textual data or any natural language processing (NLP) task, a sub-field of ML/AI dealing with text, that data first needs to be converted to a vector of numerical data by a process known as vectorization. TF-IDF vectorization involves calculating the TF-IDF score for every word in your corpus relative to that document and then putting that information into a vector (see image below using example documents “A” and “B”). Thus each document in your corpus would have its own vector, and the vector would have a TF-IDF score for every single word in the entire collection of documents. Once you have these vectors you can apply them to various use cases such as seeing if two documents are similar by comparing their TF-IDF vector using cosine similarity.

Random Forest Model

A random forest classifier. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Naive Bayes Model

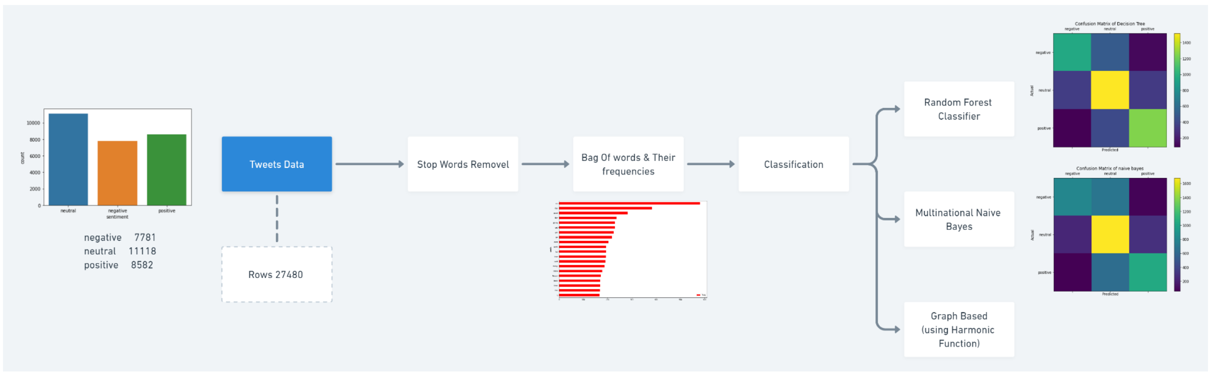
In statistics, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features . They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve high accuracy levels.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions

An approach to semi-supervised learning is proposed that is based on a Gaussian random field model. Labelled and unlabeled data are represented as vertices in a weighted graph, with edge weights encoding the similarity between instances. A new approach to semi-supervised learning that is based on a random field model defined on a weighted graph over the unlabeled and labelled data, where the weights are given in terms of a similarity function between instances.

Architecture

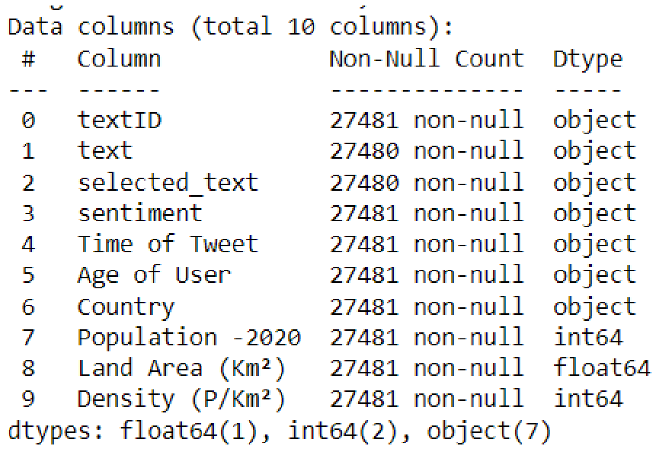

negative 7781
neutral 11118
positive 8582
Tweets Data
Rows 27480
Stop Words Removel
Bag Of words & Their frequencies
Classification
Random Forest Classifier
Multinational Naive Bayes
Graph Based (using Harmonic Function)
Confusion Matrix of Decision Tree
Confusion Matrix of naive bayes

Source: Link

Implementation

Data Collection

Data was collected from Kaggle which has several features listed below.

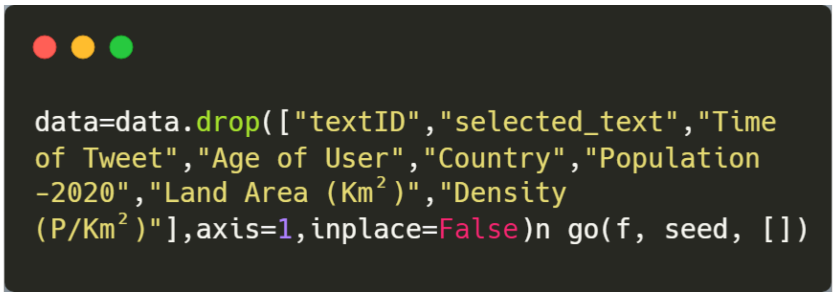


```
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   textID            27481 non-null  object 
 1   text              27480 non-null  object 
 2   selected_text     27480 non-null  object 
 3   sentiment         27481 non-null  object 
 4   Time of Tweet     27481 non-null  object 
 5   Age of User       27481 non-null  object 
 6   Country           27481 non-null  object 
 7   Population -2020  27481 non-null  int64  
 8   Land Area (Km²)   27481 non-null  float64
 9   Density (P/Km²)   27481 non-null  int64  
dtypes: float64(1), int64(2), object(7)
```

Data Cleansing

Data cleaning entails identifying incomplete, incorrect, inaccurate, or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

Here, we are dropping unnecessary columns from the dataset.



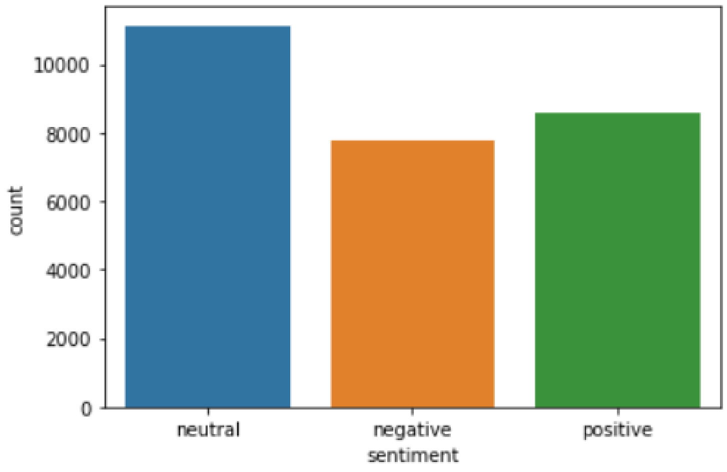
```
data=data.drop(["textID","selected_text","Time
of Tweet","Age of User","Country","Population
-2020","Land Area (Km²)","Density
(P/Km²)"],axis=1,inplace=False)n go(f, seed, [])
```

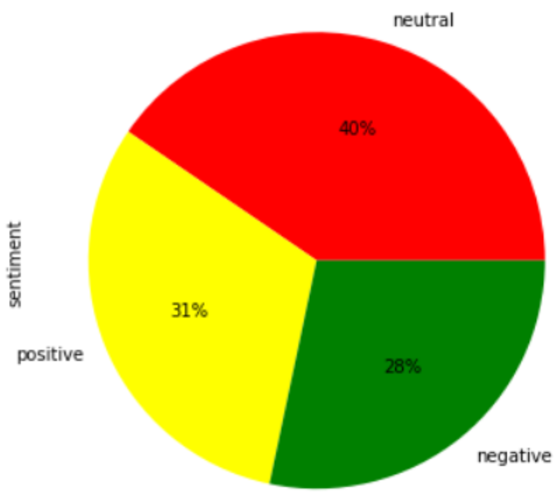
Final Data after Cleansing:

	text	sentiment	tweet_clean
0	I`d have responded, if I were going	neutral	id responded going
1	Sooo SAD I will miss you here in San Diego!!!	negative	sooo sad miss san diego
2	my boss is bullying me...	negative	boss bullying
3	what interview! leave me alone	negative	interview leave alone
4	Sons of ****, why couldn`t they put them on t...	negative	sons couldnt put releases already bought

Data Visualization

This graph categorises the tweets as positive, negative and neutral.



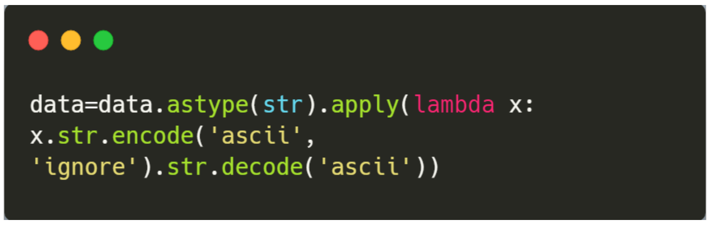


Data Preprocessing

Data quality and representativity are the primary and most significant criteria for building good forecasting models. The data preparation phase has a big impact on a machine learning algorithm's ability to generalise. Data preparation includes missing data imputation, removing or modifying outlier observations, data transformation (usually normalisation and standardisation), and feature engineering. While the first two steps are critical for collecting more exact and complete data sets, the third phase is frequently utilised to obtain data that is more evenly distributed and reduce data variability. The fourth phase is then used to create a new dataset that is typically smaller and more informative. The last stage usually includes feature extraction and feature selection.

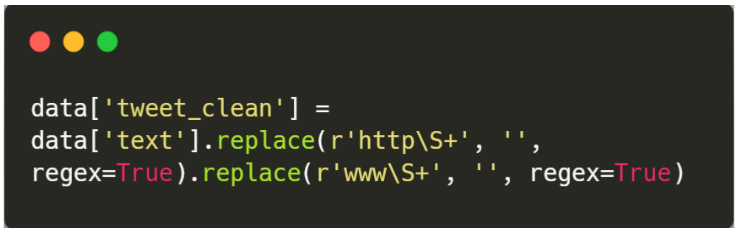
Main Functions

Remove emojis from the dataset:



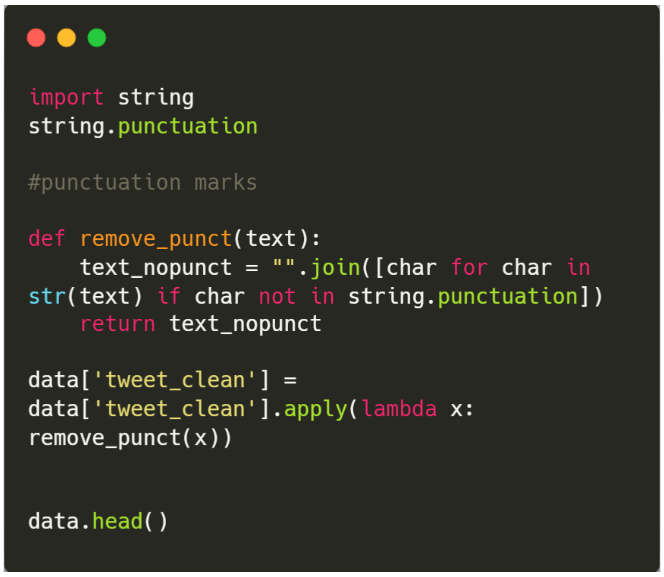
```
data=data.astype(str).apply(lambda x:
x.str.encode('ascii',
'ignore').str.decode('ascii'))
```

Remove URLs from the dataset:



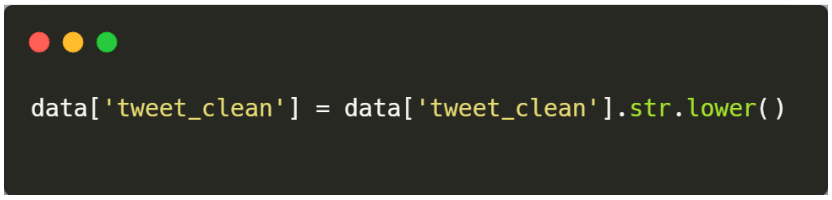
```
data['tweet_clean'] =
data['text'].replace(r'http\S+', '',
regex=True).replace(r'www\S+', '', regex=True)
```

Remove Punctuations from the dataset:



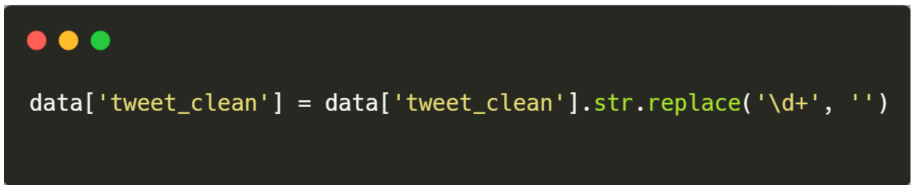
```
import string
string.punctuation

#punctuation marks

def remove_punct(text):
    text_nopunct = "".join([char for char in
str(text) if char not in string.punctuation])
    return text_nopunct

data['tweet_clean'] =
data['tweet_clean'].apply(lambda x:
remove_punct(x))


data.head()
```

Turning data into lowercase:



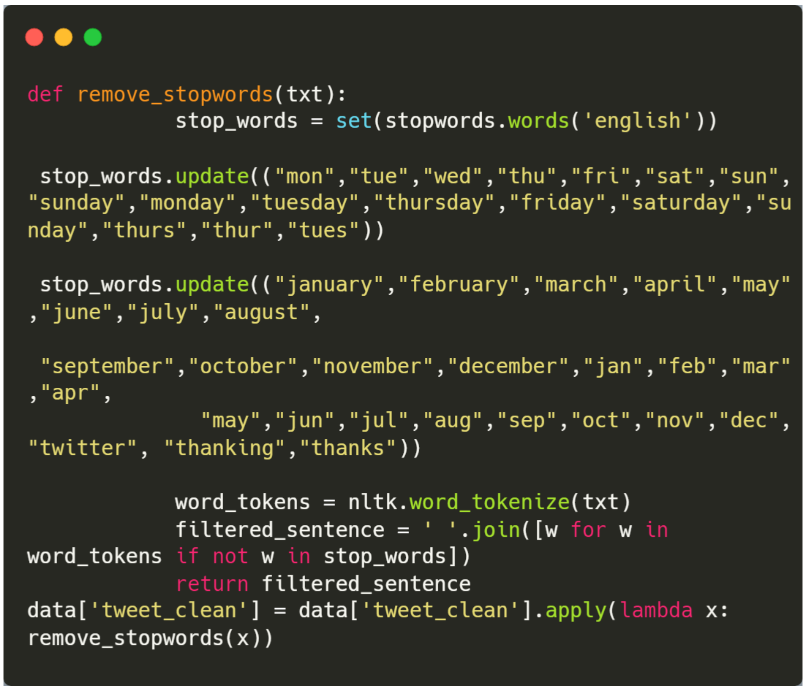
```
data['tweet_clean'] = data['tweet_clean'].str.lower()
```

Remove Numbers from dataset:



```
data['tweet_clean'] = data['tweet_clean'].str.replace('\d+', '')
```

Removing StopWords from the dataset:

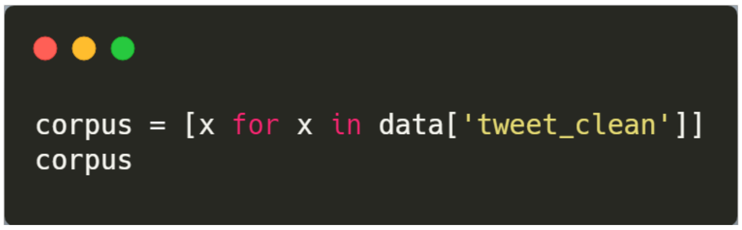


```
def remove_stopwords(txt):
            stop_words = set(stopwords.words('english'))

 stop_words.update(("mon","tue","wed","thu","fri","sat","sun",
"sunday","monday","tuesday","thursday","friday","saturday","su
nday","thurs","thur","tues"))

 stop_words.update(("january","february","march","april","may"
,"june","july","august",

 "september","october","november","december","jan","feb","mar"
,"apr",
              "may","jun","jul","aug","sep","oct","nov","dec",
"twitter",  "thanking","thanks"))

            word_tokens = nltk.word_tokenize(txt)
            filtered_sentence = ' '.join([w for w in
word_tokens if not w in stop_words])
            return filtered_sentence
data['tweet_clean'] = data['tweet_clean'].apply(lambda x:
remove_stopwords(x))
```

Final Dataset:

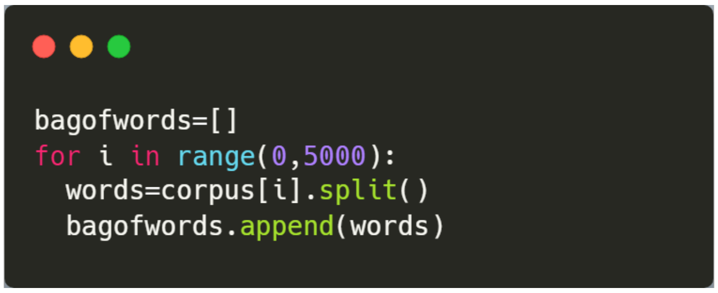
	text	sentiment	tweet_clean
0	I`d have responded, if I were going	neutral	id responded going
1	Sooo SAD I will miss you here in San Diego!!!	negative	sooo sad miss san diego
2	my boss is bullying me...	negative	boss bullying
3	what interview! leave me alone	negative	interview leave alone
4	Sons of ****, why couldn`t they put them on t...	negative	sons couldnt put releases already bought

Creating list of tweets:



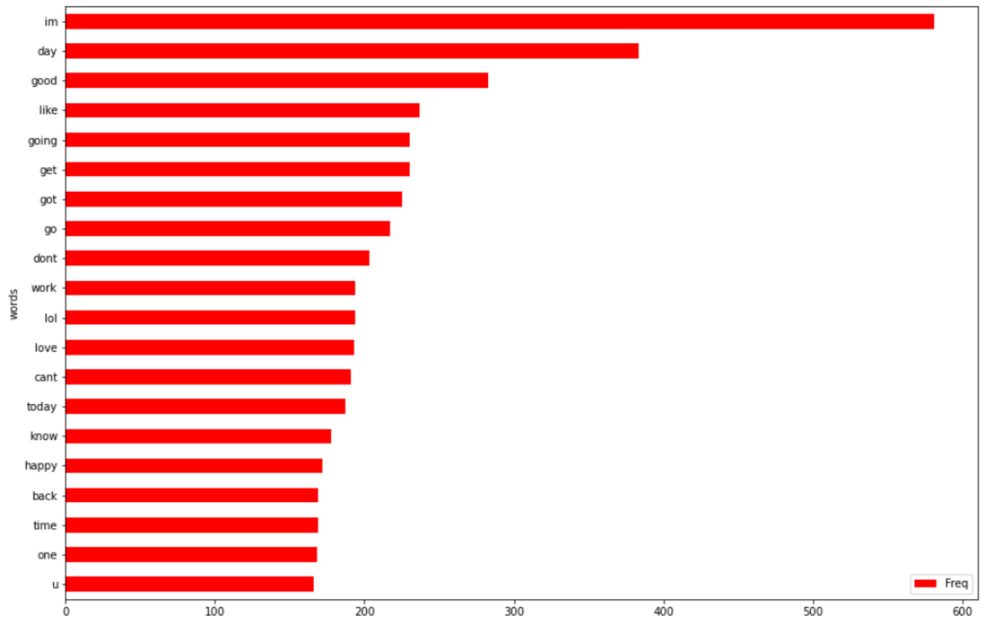
```
corpus = [x for x in data['tweet_clean']]
corpus
```

Creating Bag of words and then calculating frequency of each words:



```
bagofwords=[]
for i in range(0,5000):
  words=corpus[i].split()
  bagofwords.append(words)
```

This graph tells about the frequency of each word.



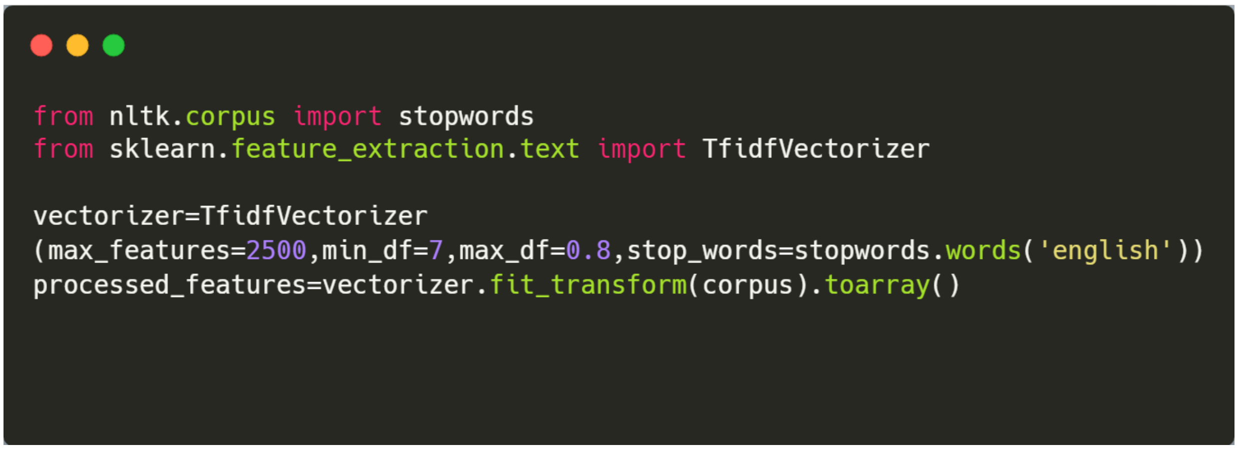
Tf-Idf Vectorizer:

Tf= (Frequency of a word in the doc) / (Total words in the document)

Idf= Log(Total number of doc)/(Number of docs containing the word)

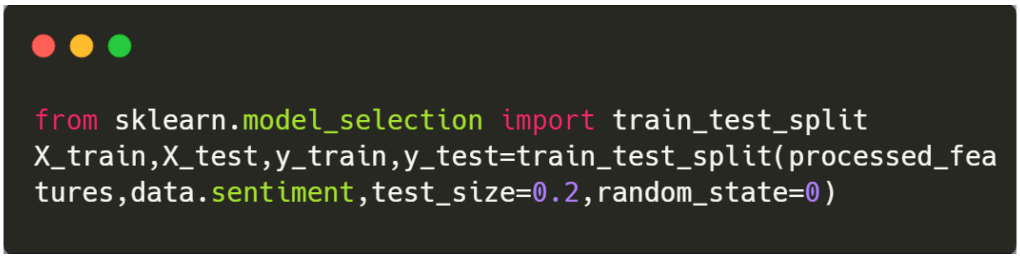
In the code below, we define that the max_features should be 2500, which means that It only uses the 2500 most frequently occurring words to create a bag of words feature vector. Words that occur less frequently are not very useful for classification.

Similarly, max_df specifies that only use those words that occur in a maximum of 80% of the documents. Words that occur in all documents are too common and are not very useful for classification. Similarly, min-df is set to 7 which shows that include words that occur in at least 7 documents.



```
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer=TfidfVectorizer
(max_features=2500,min_df=7,max_df=0.8,stop_words=stopwords.words('english'))
processed_features=vectorizer.fit_transform(corpus).toarray()
```

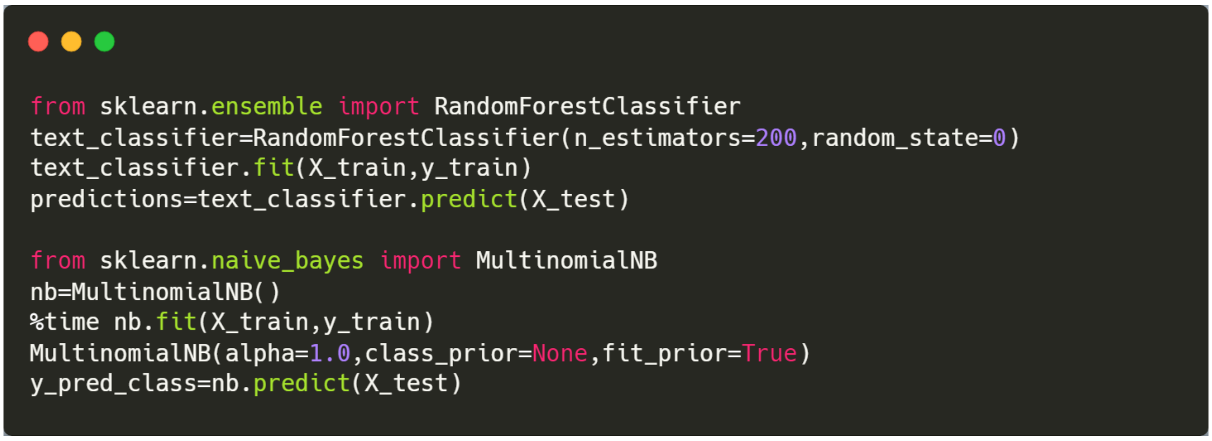
Training-Testing Dataset Formation:



```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(processed_fea
tures,data.sentiment,test_size=0.2,random_state=0)
```

Model Building:

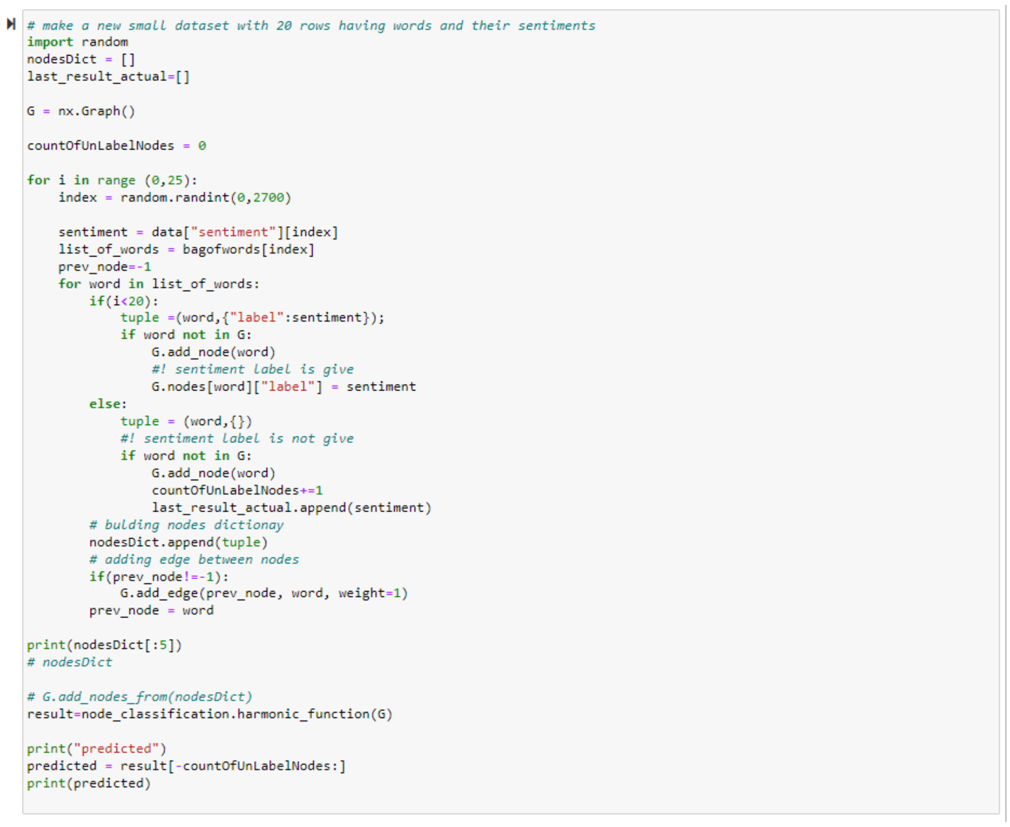
The sklearn.ensemble module contains the RandomForestClassifier class that can be used to train the machine learning model using the random forest algorithm and similarly for Navie_bayes. To do so, we need to call the fit method for both Classifier classes and pass it our training features and labels, as parameters. Then, We are calculating the confusion matrix and Accuracy Score.



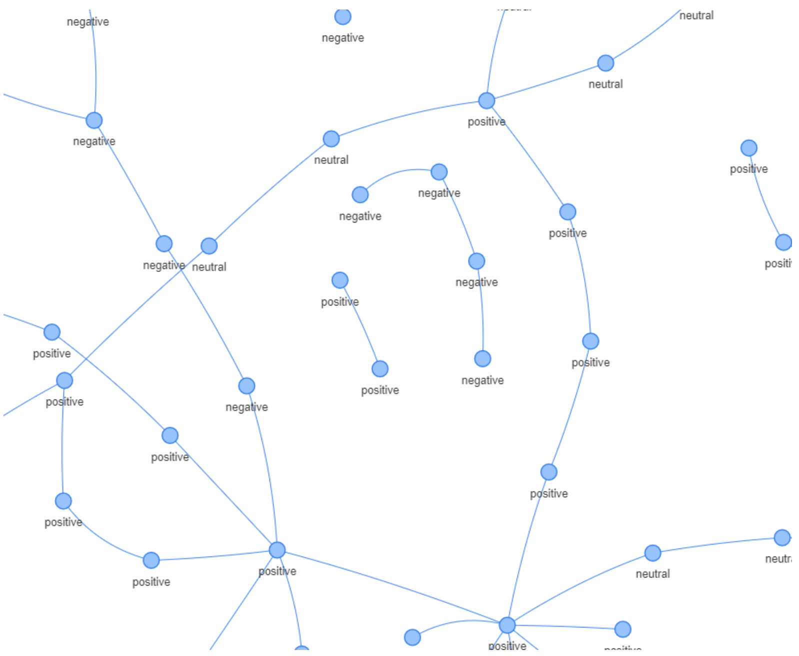
```python
from sklearn.ensemble import RandomForestClassifier
text_classifier=RandomForestClassifier(n_estimators=200,random_state=0)
text_classifier.fit(X_train,y_train)
predictions=text_classifier.predict(X_test)

from sklearn.naive_bayes import MultinomialNB
nb=MultinomialNB()
%time nb.fit(X_train,y_train)
MultinomialNB(alpha=1.0,class_prior=None,fit_prior=True)
y_pred_class=nb.predict(X_test)
```

Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions

An approach to semi-supervised learning is proposed that is based on a Gaussian random field model. Labelled and unlabeled data are represented as vertices in a weighted graph, with edge weights encoding the similarity between instances.



```python
# make a new small dataset with 20 rows having words and their sentiments
import random
nodesDict = []
last_result_actual=[]

G = nx.Graph()

countOfUnLabelNodes = 0

for i in range (0,25):
    index = random.randint(0,2700)

    sentiment = data["sentiment"][index]
    list_of_words = bagofwords[index]
    prev_node=-1
    for word in list_of_words:
        if(i<20):
            tuple =(word,{"label":sentiment});
            if word not in G:
                G.add_node(word)
                #! sentiment label is give
                G.nodes[word]["label"] = sentiment
        else:
            tuple = (word,{})
            #! sentiment label is not give
            if word not in G:
                G.add_node(word)
                countOfUnLabelNodes+=1
                last_result_actual.append(sentiment)
        # bulding nodes dictionay
        nodesDict.append(tuple)
        # adding edge between nodes
        if(prev_node!=-1):
            G.add_edge(prev_node, word, weight=1)
        prev_node = word

print(nodesDict[:5])
# nodesDict

# G.add_nodes_from(nodesDict)
result=node_classification.harmonic_function(G)

print("predicted")
predicted = result[-countOfUnLabelNodes:]
print(predicted)
```



```
[('spent', {'label': 'negative'}), ('hours', {'label': 'negative'}), ('looking', {'label': 'negative'}), ('blog', {'label':
'negative'}), ('topic', {'label': 'negative'})]
predicted
['positive', 'negative', 'negative', 'negative', 'negative', 'negative', 'negative', 'negative', 'negative', 'positive', 'po
sitive', 'neutral', 'neutral', 'neutral', 'neutral', 'negative', 'negative', 'negative', 'negative', 'negative', 'negative',
'negative', 'negative']
```

Results

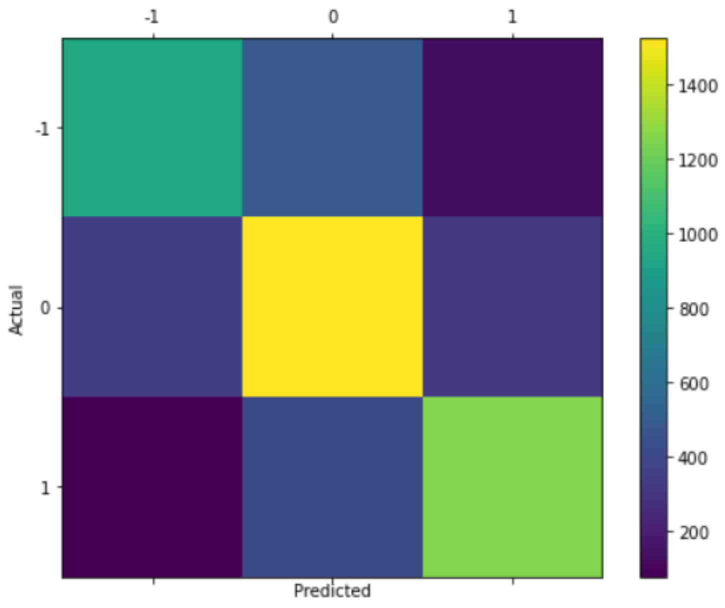
Random Forest:

```python
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
```

```
[[ 947  491  125]
 [ 346 1528  321]
 [  78  401 1260]]
              precision    recall  f1-score   support

    negative       0.69      0.61      0.65      1563
     neutral       0.63      0.70      0.66      2195
    positive       0.74      0.72      0.73      1739

    accuracy                           0.68      5497
   macro avg       0.69      0.68      0.68      5497
weighted avg       0.68      0.68      0.68      5497

0.6794615244678915
```

Confusion Matrix:

```
[[ 947  491  125]
 [ 346 1528  321]
 [  78  401 1260]]
```



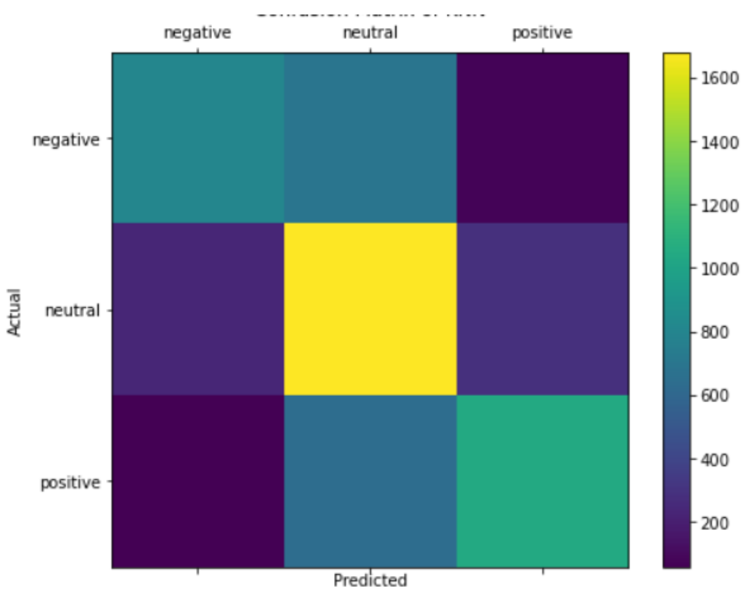
Naive Bayes:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
print(confusion_matrix(y_test,y_pred_class))
print(classification_report(y_test,y_pred_class))
print(accuracy_score(y_test,y_pred_class))
```

```
[[ 804  687   72]
 [ 232 1680  283]
 [  58  640 1041]]
              precision    recall  f1-score   support

    negative       0.73      0.51      0.61      1563
     neutral       0.56      0.77      0.65      2195
    positive       0.75      0.60      0.66      1739

    accuracy                           0.64      5497
   macro avg       0.68      0.63      0.64      5497
weighted avg       0.67      0.64      0.64      5497

0.6412588684737129
```

Confusion Matrix:

```
[[ 804  687   72]
 [ 232 1680  283]
 [  58  640 1041]]
```



Conclusion

From the above result, we can conclude ,Random Forest is best among the two models to detect the depressive tweets.

References

[1] Sumandeep Kaur, Geeta Sikka, and Lalit Kumar Awasthi ,Sentiment Analysis Approach Based on N-gram and KNN Classifier,2018
[2] Dipti Mahajan; Dev Kumar Chaudhary, Sentiment Analysis Using Rnn,2018
[3] Xi Ouyang; Pan Zhou; Cheng Hua Li; Lijun Liu, Sentiment Analysis Using Convolutional Neural Network,2015
[4] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML'03). AAAI Press, 912–919.